# Service composition with Partial Goal Satisfaction*

Luca Sabatucci, Massimo Cossentino, Salvatore Lopes

National Research Council of Italy (CNR)
Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
Via U. La Malfa, 153, Palermo (Italy)
{[name].[surname]}@icar.cnr.it
http://www.ecos.pa.icar.cnr.it

**Abstract.** IoT applications are often ad-hoc compositions of services offered by connected devices that cooperate to satisfy user's goals. Sometimes, addressing full goal satisfaction is too stringent and replacing that with an easier to satisfy partial goal satisfaction is a good alternative to a complete failure. In this paper we propose a service composition approach that adopts a metrics for measuring the partial satisfaction of goal. The metrics adopts an electrical analogy extended for dealing with temporal goals.

## 1 Introduction

Today, the Internet of Thing (IoT) plays a more and more important role in our lives. According to a recent report from Cisco, the IoT will consist of 500 billion devices connected to the Internet by 2030. Each device includes sensors that collect data, interact with the environment, and communicate over a network. IoT applications are built as emerging compositions of devices' services according to contingent user's goals[3]. However, in realistic environments, it is important to consider scenarios where the full realization of a goal is not always possible. Service unavailability and lack of resources may hinder the full satisfaction whereas some parts of the user's specification could be still achievable and at least partially satisfy user expectations.

The research topic of *partial goal satisfaction* yields to overcome the barrier of utilizing existing planners when boolean satisfaction conditions are too strict. Researchers study mechanisms for differentiating between optimal and feasible plans, thus, in case the optimal plan does not exist, the planner may still return something more than the *empty plan*. In literature, there are many application domains that lead to different proposal of partial satisfaction, often depending on the specific definition of goals. In some domains, variables are continuous, and a goal is defined as the maximization/minimization of an utility value representing how much the goal worths for the user. In these cases, partial satisfaction means

* Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

reaching a non-optimal value. This may often happens when reasoning with incomplete information about the context [8, 7].

In many cases, the goal model is defined as a tree of goals, obtained through AND/OR decompositions of a root goal. Here, partial satisfaction means relaxing some of the goals thus to be dealt as weak constraints. In this view, a valid plan may fulfil only a subset of them. For instance, [2] adopts a cost-sensitive reachability heuristics by associating benefits to goal satisfaction and costs to actions. In formal logical approaches, goals are expressed as predicate formulas (for instance: $g = x \wedge y$). Asserting $g$ is partially satisfied requires to re-define the meaning of the classical implication operator. This approach has been followed by Zhou et al. [10], in the context of background knowledge, in which they introduce a family of partial implication operators. In [9] the authors propose an approach for dealing with goals as a conjunction of states and partial satisfaction corresponds to the fulfilment of some of these states. Goals are divided in core goals (representing user's interests) and context goals (attributes of the previous ones). The authors provide a way for transforming failed goals into problems that can be solved by using an AI planner. The problem also arises when goals are expressed via temporal logic. In [6], authors accept the compromise to violate the goal for a small time interval in order to obtain a fulfilment later.

In this paper we propose a service composition approach that adopts a metrics for measuring the partial satisfaction with respect to temporal goals. Our idea of partial goal satisfaction has been already stated in [5]. This paper presents two novelties:

1. The partial goal satisfaction has been exploited in the context of a planner for service composition: it provides the heuristics for evaluating all the visited states, thus selecting the most promising for addressing the final goal.
2. The concept of Resistance to Goal Satisfaction is extended to deal with some temporal operators: Finally and Globally.

The paper is structured as follows: Section 2 presents the framework used for service composition. Section 3 describes the resistance to goal satisfaction (R2S) metrics for measuring the partial goal satisfaction of a propositional formula. Section 4 extends the R2S formulation for the Finally and the Globally operators by adopting Petri-net models. The final Section 5 reports some conclusions and future works.

## 2 Service Composition by Planning

The service composition domain is presented by means of a non-deterministic state-transition system, namely the World Transition System built for describing the effects of actions, in order to address a specified goal.

**Definition 1 (World Transition System).** *Given $\Sigma$, the alphabet of predicates, and $C$ the set of capabilities, the World Transition System (WTS) is a tuple $WTS =< W, T, R, L >$ where:*

– *$W \subseteq 2^{\Sigma}$ is the set of states of the world*
  *(where $\boldsymbol{w} \in W$ is a vector of boolean variables, associated to $\Sigma$)*
– *$T : W \times C \to W$ is the transition relation*
– *$R : W \to \mathbb{R}$ is the scoring function*
– *$L : W \to \{S, F, WS, WF\}$ is the temporal labeling function*

A capability $c \in C$ represents the non deterministic action associated to a service. It wraps the service by denoting a precondition (the condition necessary for executing the service) and a range of possible effects. Each capability effect describes one possible state evolution $W \to W$ due to the service invocation.

In this context, the planning domain is identified by the tuple $D =< I, C_{av}, G >$ where:

– $w_I \in W$ is the initial state of the world,
– $C_{av} \subseteq C$ is a set of capabilities that are available at the time of planning,
– $G$ is a set of goals expressed in linear temporal logic (LTL).

The Linear Temporal Logic (LTL) [4] is often used in software engineering for the formal specification of goals and system properties [1]. It extends the propositional logic to model *the future* by using an infinite sequence of states that represents discrete moments of time (every state has only one successor).

**Definition 2 (Linear Temporal Logic).** *The LTL goal $\phi$ is defined as follows:*

$$\phi \equiv p \mid \top \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid G\phi \mid F\phi \mid \phi U \phi \mid X\phi \tag{1}$$

*where $p \in \Sigma$ are atomic formulas, i.e. simple predicates.*

Briefly, LTL formulas are based on propositions and their combination through logical connectors and temporal operators. While logic connectors can only describe a static behaviour, limiting their arguments validation to only one state, temporal operators requires looking at multiple states for their validation. The basic and derived temporal operators are:

– The X operator (Next) imposes that $\varphi$ will be true in the next state.
– The U operator (Until) states that $\varphi$ will be true in every state until $\psi$ becomes true and imposes that sooner or later $\psi$ will become true.
– The F operator (Finally, sometimes $\diamond$) $F\phi = \top U \phi$ states that eventually, at least once, $\varphi$ will be true in a future state.
– The G operator (Globally, sometimes $\Box$) $G\phi = \neg F \neg \phi$ means $\varphi$ must be true in every state (current and future).

The WTS is non-deterministic because the same capability may be associated to many world evolutions. In practice, at planning time, these evolutions are alternative, and the planner must consider all of them. Indeed, the result of the planning encompasses plans with decision points and loops.

**Definition 3 (Non-Deterministic Solution).** *A Non-deterministic Solution $\pi$, is defined inductively as $\pi = c \mid c \cdot \pi \mid \pi \diamond (\pi \otimes \pi)$ where $c \in C_{av}$, $\cdot$ is the concatenation operator, $\diamond$ is the decision operator and $\otimes$ is the exclusive or.*

Intuitively, a plan $\pi$ identifies a workflow of actions. The definition of $\pi$ includes sequences (classical plans), decision points (non-deterministic plans) and loops, as shown in Figure 1. Whereas classical planning produces plans which execution will address the given goal, in this paper, we front a slightly different problem. We search for plan which execution may not lead to the desired final state.

**Definition 4 (Full/Partial Solution).** *A plan $\pi$ may lead to either Full or Partial satisfaction of the given goal $\phi$:*

- *$\pi$ is a **full solution** when the execution of the plan $Exec(\pi) \models \phi$;*
- *otherwise, the plan $\pi$ is said **partial solution**.*

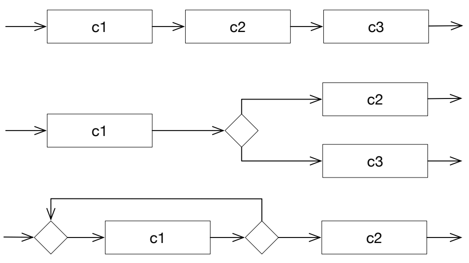


**Fig. 1.** The solution of the planner is a workflow. It can incorporate three patterns: sequences $\pi = c_1 \cdot c_2 \cdot c_3$, decision points $\pi = c_1 \diamond (c_2 \otimes c_3)$ and loops $\pi = c_1 \diamond (\pi \otimes c_3)$.

The proposed planning algorithm for service-composition with partial goal satisfaction derives from a *best-first search* in the state-of-world state.

```
Data: $w_0$,$G$,$cap\_set$
Result: $solutions$
Function service_composition($w_0$,$G$,$cap\_set$):
    $solutions \leftarrow \emptyset$
    $WTS \leftarrow initialize\_WTS(w_0)$
    repeat
        $node_i \leftarrow get\_highest\_from\_frontier(WTS)$
        $expansion\_set \leftarrow service\_composition(node_i, cap\_set)$
        foreach $state \in expansion\_set$ do
            $state.token \leftarrow check\_LTL(state, goal, node_i)$
            if $state.token = SUCCESS$ then
                $solutions \leftarrow search\_solutions(WTS, state)$
            else
                $score \leftarrow evaluate\_partial\_sat(state, token)$
                $WTS \leftarrow update\_WTS(WTS, state, token, score)$
            end
        end
    until $size(solutions) \leq MAX\_SOL$ and $size(WTS) \leq MAX\_SPACE$
    $partial\_solutions \leftarrow search\_partial(WTS)$
    $sort(partial\_solutions)$
End Function
```

**Algorithm 1:** Planning Algorithm for Service Composition with Partial Goal Satisfaction

Algorithm 1 iteratively builds a WTS, starting from the initial state $w_0$. At each step the algorithm looks at the frontier (the nodes that are not yet visited) and it expands the most promising node by using the applicable capabilities. The preference on the node to be selected for the expansion is evaluated using some user heuristic. In this planning algorithm, temporal aspects are useful for determining full and partial solutions. Each node is marked with four types of temporal markers: full success (S), total failure (F), partial success (WA) and partial failure (WF). These are explained in details in Section 4.

Indeed, $\pi \subset WTS$ is a full solution when it starts from the initial state $w_0$, it does not contain any Failure state, and all the subpaths terminate in a Success state (loops are valid if they have an exit condition towards a Success state). Typically the algorithm terminates when a sufficient number of full solutions are discovered. However, it may be the case that such solutions do not exist. In this case the algorithm terminates when the WTS size is over a given threshold[1]. When the algorithm terminates, we explore the WTS for extracting partial solutions. $\pi \subset WTS$ is a partial solution when the initial state is $w_0$, but (differently from full solutions) it can contain some Failure states, and subpath that terminates in a partial success state (WS) are allowed. Clearly, not all the partial solutions are equally acceptable. We defined a metrics for ordering them according to a "Distance to Goal Satisfaction". This metrics will be discussed in details later, in subsection 3.1.

## 3 Measuring the Distance to Satisfaction

A goal is a logical formula composed by propositional variables, logical connectives and temporal operators, as seen in Section 2.

[1] Alternatively, Algorithm 1 may terminate after a given time interval.

An example of goal may be the specification of the status of some traffic lights and direction signals for speeding up the evacuation of inhabitants in a city district in case of an emergency.

In our study we consider the possibility that some of the goal conditions cannot be achieved by the system itself without the intervention of the user. For instance, some traffic direction signals may not be activated by a remote system and therefore a human being has to correctly deploy them.

### 3.1 The R2S Metrics for the Propositional Logic

In some situations, goals cannot be achieved even involving the human in their pursuit. Therefore we explicitly consider the possibility to finish our solution planning activity with a Not(Goal) condition that means the intended goal formula has not been verified. Usually, by employing the available services in one of the studied plans, the system may reach states where some of the variables composing the goal formula are satisfied. We call these *Partial Solution States* and, the problem becomes to find a metrics for selecting the best partial solution state. Even a state where none of the propositional variables in the goal formula is verified may be seen as a partial solution state, of course that would be the least preferable one.

In considering what metrics could be useful for evaluating the goodness of a partial solution state, we could refer to contributions from literature like [9] but we now prefer states that are in a strategic position (in the solution space) for reaching the full satisfaction. This may be a little different from preferring a state that maximizes some kind of utility (or quality of service) function since the latter would be a good state for an indefinite positioning of the systems but it could also not be a well-suited state for later achieving the full goal satisfaction.

Going back to the example of traffic lights and direction signals, suppose the best evacuation route may not entirely ruled by traffic lights and direction signals because one turn at a specific point is not provided with any remote-controlled device. Two options of partial satisfaction may be pursued: in the first the system chooses an alternative (sub-optimal) route, in the second the system activates the best route and warns the user of the lack of a signal in a specific location. A police unit may be dispatched to solve that in order to obtain the full satisfaction. Which of the two solutions is preferable strongly depends on the problem and on the specific emergency (system status). For this reason, in many applications we suggest leaving the final decision to the human. It is worth noting that adopting the first option (good state for staying in a partial solution situation) may bring the system far from a state the ensures a viable solution at the next step (for instance with a human intervention).

In order to measure the distance of a state in the solution space from the desired goal we introduce the following definition [5]:

**Definition 5 (Distance to Goal Satisfaction).** *The Distance to Goal Satisfaction (DGS) of a state, in the world-state space, is the distance –in terms of the number of variables to be changed, and the degree of freedom in changing them– of this state towards the nearest one that satisfies the goal.*

According to this informal definition, the Distance to Goal Satisfaction depends on two factors:

– The number of variables to be changed: this is the minimum number of propositional variables in the goal formula that have to change their value in order to fully satisfy the goal.
– The degree of freedom: sometimes the goal formula may be satisfied by changing the value of n variables chosen among m candidate variables (with m>n). An obvious case for this situation is represented by the OR condition where it is possible to verify the OR formula by changing to true the value of one of the two variables.

Intuitively we may justify this approach by saying that the more variables it is necessary to change in order to satisfy the goal, the more the point in the solution space is far from the point representing the goal. Indeed, this is not sufficient in the proposed approach: two different points with the same number of required changes have different proximity to the goal if one of them offers the possibility to choose the variables to be changed in a large number of options.

In order to operationalize the estimation of the Distance to Goal Satisfaction, we introduce a metrics, called Resistance to Goal Satisfaction that adopts an electric analogy for deducing an electric circuit from the goal formula. The equivalent resistance offered by this circuit will be considered as an estimator of the distance to goal satisfaction.

## 3.2 Electrical Analogy: from Goal Formula to Equivalent Circuit

The transformation of the goal formula to the equivalent circuit is based on some simple rules:

– Each logic variable of the goal formula is represented by a resistor in the equivalent circuit.
– The value of each resistor depends on the value of the logic variable. If that is true, the value will be $R_{min}$, otherwise it will be $R_{max}$. Such values are chosen very different in magnitude, ideally they would be $R_{min}= 0$ (short circuit) for the true variable, $R_{max}= \infty$ (open circuit) for the false variable but this would create problems in calculating the equivalent resistance for some circuits.
– Each AND operator corresponds to the series connection of the resistors representing the variables in the AND conjunction.
– Each OR operator corresponds to the parallel connection of the resistors representing the variables in the OR disjunction.
– Each NOT operator corresponds to inverting the value of the resistor representing the negated variable (if the variable is true, the corresponding resistor value is $R_{max}$ and so on).

This analogy allows calculating R2S of propositional formulas, as shown in Algorithm 2.

```
**Data:** $\phi$,$w_i$,$R = \{R_A, R_B, ...\}$
**Result:** $R2S(\phi)$
**Function** r2s($\phi$,$w_i$,$R = \{R_A, R_B, ...\}$):
    **case** $\phi = p$ *is a predicate* **do**
        **if** $w_i \models p$ **then**
            **return** $R_p$
        **else**
            **return** $1/R_p$
        **end**
    **case** $\phi = \alpha \wedge \beta$ **do**
        **return** $R2S(\alpha) + R2S(\beta)$
    **case** $\phi = \alpha \vee \beta$ **do**
        **return** $1/(1/R2S(\alpha) + 1/R2S(\beta))$
    **case** $\phi = \neg\alpha$ **do**
        **return** $1/R2S(\alpha)$
    **end**
**End Function**
```

**Algorithm 2:** Function for calculating the R2S of propositional formulas.

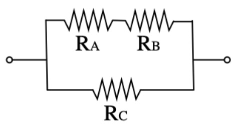
It is worth to note that different values may be chosen for the resistance representing each propositional variable. This would correspond to considering a different relevance for the variable in the goal formula. We are not studying this case since it would bind the R2S to the domain and we prefer to maintain that as general as possible.

*Property 1 (De Morgan's Laws).* The R2S metrics is valid with respect the following transformations:
$R2S(\neg(A \cup B)) = R2S(\neg A \cap \neg B)$ and
$R2S(\neg(A \cap B)) = R2S(\neg A \cup \neg B)$.
also known as De Morgan's Laws.

This property may be easily verified by applying the above reported algorithm. Now, let us suppose we want to study the following goal formula:

$$\mathrm{G} = (\mathrm{A} \wedge \mathrm{B}) \vee \mathrm{C}$$

According to the proposed transformation rules, the equivalent circuit and the equivalent resistance R2S would be the ones represented in Fig. 2.



$$R2S = \frac{(R_A + R_B) \cdot R_C}{R_A + R_B + R_C}$$

**Fig. 2.** The equivalent circuit and the corresponding equivalent resistance R2S

The goal G will be verified for five conditions of the A,B,C variables as reported in the table in Fig.3.

The columns of the table reports: the values of A,B,C, the corresponding value of the goal G, the minimum number of variable changes that are necessary

| A | B | C | G | Num. Changes | Freedom | R2S |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | $\frac{2}{3}R_X$ |
| 0 | 0 | 1 | 1 | - | - | 0 |
| 0 | 1 | 0 | 0 | 1 | 2 | $\frac{1}{2}R_X$ |
| 0 | 1 | 1 | 1 | - | - | 0 |
| 1 | 0 | 0 | 0 | 1 | 2 | $\frac{1}{2}R_X$ |
| 1 | 0 | 1 | 1 | - | - | 0 |
| 1 | 1 | 0 | 1 | - | - | 0 |
| 1 | 1 | 1 | 1 | - | - | 0 |

**Fig. 3.** Values of R2S for the different conditions of the A,B,C variables

in order to satisfy G, the degree of freedom in variable changes, the resistance to satisfaction R2S.
Three different conditions do not verify the goal, they are: (0,0,0), (0,1,0), (1,0,0). The aim of the proposed metrics is to estimate which one would be the best choice for our planner. By adopting the equivalent resistance formula reported in Fig.2, and supposing we adopt the value $R_X$ for $R_{max}$ and the value $R_m$ for $R_{min}$ where the previous one is several orders of magnitude bigger than the second one ($R_X >> R_m$) we obtain the following result for the R2S for the condition (0,0,0):

$$R2S(0,0,0) = \frac{(R_A + R_B)R_C}{R_A + R_B + R_C} = \frac{(R_X + R_X)R_X}{R_X + R_X + R_X} = \frac{2}{3}R_X \qquad (2)$$

While the R2S for the condition (0,1,0) is:

$$R2S(0,1,0) = \frac{(R_A + R_B)R_C}{R_A + R_B + R_C} = \frac{(R_X + R_m)R_X}{R_X + R_m + R_X} \approx \frac{1}{2}R_X \qquad (3)$$

The R2S for the last condition of partial goal satisfaction (1,0,0) may be easily calculated as the previous one (with the same result).

Finally, in order to verify the result provided by the proposed approach in case of full goal satisfaction, we can calculate the R2S for the (0,0,1) condition:

$$R2S(0,0,1) = \frac{(R_A + R_B)R_C}{R_A + R_B + R_C} = \frac{(R_X + R_X)R_m}{R_X + R_X + R_m} \approx R_m \approx 0 \qquad (4)$$

As it can be seen, the proposed metrics successfully discriminate among conditions where the same number of variable changes are needed but different degrees of freedom are available. In fact while all the conditions (0,0,0), (0,1,0), and (1,0,0) need one variable change to satisfy G, only the last two ones allow to achieve that by choosing the one variable between two options. Corresponding values of R2S are minor in the last situation thus proposing to the planner to firstly explore points of the solution space that have this value.

## 4 Petri-Nets for Temporal Operators

The proposed approach uses an extended notation of a Petri-net where places and transitions are semantically annotated. Due to the lack of space, this section only aims at giving an informal idea of the approach. In semantically annotated Petri-nets, transitions may be annotated with propositional conditions. The annotated transition fires iff (if and only if) all the input places contain a token and the current state-of-the-world satisfies the condition. At every discrete time instant only 1 transition may fire, and the choice depends on the current state of the world.

In addition, Places may be associated to a "success indicator" and to a R2S value. The "success indicator" may be:

- *Accepting* (A) meaning that the goal is fully addressed; an example is given when $\phi = Fa$ and the current state satisfies 'a'.
- *Waiting, but Accepted* (WA) in which the goal is not totally addressed but there are not violations. Consequently, if the system terminates, it concludes with an acceptance. An example occurs when the goal is $\phi = Ga$ and the path always satisfies 'a': we can never conclude with a success until the system terminates (because future events may lead to violations).
- *Failure* (F) meaning that a violation of the goal happened. An example is given with $\phi = Ga$ and '$\neg$a' occurs.
- *Waiting, but Failure* (WF) in which the goal is not yet addressed and if the system terminates, it concludes with a failure. An example is given when $\phi = Fa$ and 'a' has not yet happened.

Each LTL operator (X,U, F,G) may be translated into an annotated petri-net model with an initial token configuration. In the following we discuss only two of the most frequent cases, just to give a preliminary idea of the approach.

### 4.1 Finally

The Finally operator prescribes a condition will eventually hold. In a finite transition system, the condition must hold after a finite time instant. Until the condition does not hold, the system Wait (with Failure) to eventually move towards an Accepting state. Figure 4 shows a Petri-net with these two annotated places. In addition, there is a special counter place in which, at each step where the condition does not hold, a new token is accumulated: the number of tokens measures how long the system stays in the WF state. At each time instant, the state may be *Waiting, but Failure* or *Accepting*, and the R2S may be, respectively, calculated as $R2S = f(n)R_a$ or $1/R_a$.

When in state WF, the resistance to goal increases with the number of tokens in the counter place, thus modeling the urgency of reaching the Accepting state. The $f(n)$ function may be chosen according to the specific domain of application. On the right side of Figure 4 the $f(n)$ function is drawn as linear, however other kinds of dependency may be modelled (exponential, logarithmic) as well.

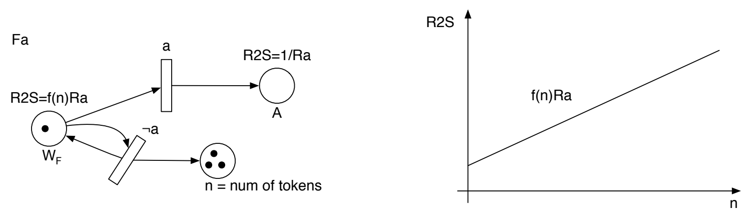


**Fig. 4.** Petri-Net pattern for the Finally operator.

## 4.2 Globally

The globally operator is generally used to represent safety properties, i.e. condition that must always hold (or undesired properties that must never happen). A finite transition system must continuously positively verify the condition (within a finite time instant) or conclude with a Failure. Figure 5 shows a petri-nets with a *Waiting, but Accepted* place and *Failure* place. Again, there is a special counter place in which, at each step where the condition holds, a new token is accumulated: the number of tokens measures how long the petri-net stays in the WF state. At each time instant, R2S may be, respectively, calculated as $R2S = g(n)R_b$ until the condition holds or $R_{max}$ after a failure.

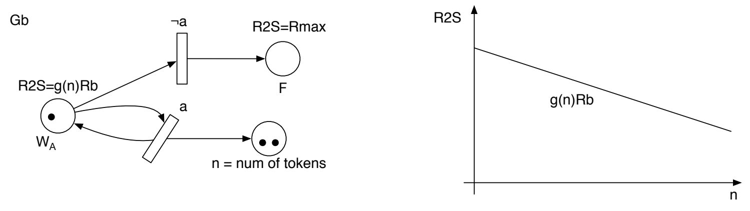


**Fig. 5.** Petri-Net pattern for the Globally operator.

When in state WA, the resistance to goal decreases with the number of tokens in the counter place, thus modeling the cumulative probability to remain in a safe state. In this case too, the $g(n)$ function may be linear as on the right side of Figure 4 or exponential, logarithmic, according to specific needs.

Algorithm 3 describes the function evaluate_partial_sat, used during the planning (see Figure 1).

```
Data: $w_i$,$PN$
Result: $R2S$
Function evaluate_partial_sat($w_i$,$PN$):
    $update\_tokens(PN, w_i)$
    return $R_{array} \leftarrow getR2S\_from\_place\_with\_token(PN)$
End Function
```

**Algorithm 3:** Algorithm for the evaluation of Partial Satisfaction with temporal formulas.

Every time the planner generates a new state, it invokes Algorithm 3. First step: the *update_tokens* function checks for all the formula petri-nets for activating transitions' firing. Second step: the *getR2S_places_with_tokens* function explores all the semantic places (A, WA, WF, F) and, for each of them, calculates the corresponding R2S. Finally, Algorithm 3 sums the resulting array of R2S; the result representes the effort yet to do for moving towards an accepting final state.

## 5 Conclusions

This paper has presented a service composition algorithm that is based on a best-first state search approach. The heuristic we adopted is the R2S, i.e. the resistance to satisfaction, inspired to an electrical analogy, in which the effort to spend for moving from the current state towards the final state is represented as a resistor. The temporal extension encompasses some Petri-Net patterns that provides variable resistances that increase/decrease with the time, thus rendering the urgency to reach the goal, or the risk to leave a safe property.

## References

1. Autili, M., Grunske, L., Lumpe, M., Pelliccione, P., Tang, A.: Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar. IEEE Transactions on Software Engineering **41**(7), 620–638 (2015)
2. Benton, J., Do, M., Kambhampati, S.: Anytime heuristic search for partial satisfaction planning. Artificial Intelligence **173**(5-6), 562–592 (2009)
3. Casadei, R., Fortino, G., Pianini, D., Russo, W., Savaglio, C., Viroli, M.: A development approach for collective opportunistic edge-of-things services. Information Sciences **498**, 154–169 (2019)
4. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems (TOPLAS) **8**(2), 244–263 (1986)
5. Cossentino M., Sabatucci L. and Lopes S.: Partial and full goal satisfaction in the musa middleware. In: Multi-Agent Systems - Prof. of the European Conference on Multi-Agent Systems (EUMAS) 2018. Lecture Notes in Computer Science book series (LNCS), vol. 11450, pp. 15–29. Springer (2018)
6. Lahijanian, M., Almagor, S., Fried, D., Kavraki, L.E., Vardi, M.Y.: This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
7. Lakner, P.: Utility maximization with partial information. Stochastic processes and their applications **56**(2), 247–273 (1995)
8. Mania, M., Santacroce, M.: Exponential utility maximization under partial information. Finance and Stochastics **14**(3), 419–448 (2010)
9. Vukovic, M., Robinson, P.: Goalmorph: Partial goal satisfaction for flexible service composition. In: Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference on. pp. 6–pp. IEEE (2005)

10. Zhou, Y., Van Der Torre, L., Zhang, Y.: Partial goal satisfaction and goal change: weak and strong partial implication, logical properties, complexity. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1. pp. 413–420. International Foundation for Autonomous Agents and Multiagent Systems (2008)